多功能蓝牙手套

无 23 尤忆晨 2022010576　无 21 李锦轩 2022010519

1. 设计背景

我们组设计的作品名为多功能蓝牙手套，设计灵感来源于我们认为市面上的 PPT 翻页笔功能太过单一，传统鼠标的使用太过局限，并且还存在一些不方便使用鼠标与键盘的场合，因此我们组设计了一个数据手套来代替鼠标和键盘的功能。

2. 功能简介

我们的多功能蓝牙手套主要有以下三个功能，首先是蓝牙键鼠，我们将手势分为默认手势以及自定义手势，其中默认手势是我们已经设置好的，如食指弯曲相当于点击鼠标左键，中指弯曲代表点击鼠标右键等等。第二个功能是手势自定义，我们用 Qt Creator 编写了一个上位机能够让用户自定义手势（如图 1），手势可以是静态的，如比 OK 手势，也可以是动态的，如打勾，用户在手势录制完成后可以在库中挑选对应的功能，如调整音量、局部截屏等等。第三个功能是实时动捕，我们用 unity3D 搭建了一个手部模型（如图 2），能够捕捉到手部的动作，并且搭建了简单的场景，能够完成拍球等简单交互。

图 1

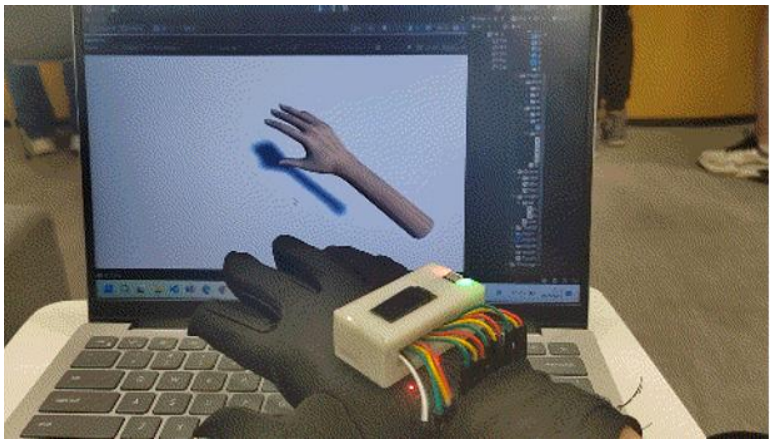

图 2

3.模块使用

为了使我们的手套更加轻便，我们使用了自制的 pcb 主板(带 oled 屏幕)，尺寸仅有 5.2cm x 2.2cm 和一个扩展 pcb 板，用于连接弯曲度传感器（如图 3-5）。在此期间我们一共打了三版板子，但前两版均因集成度不够高被淘汰，最终第三版的体积能够满足我们的要求。我们的 pcb 板上搭载了一个 esp32 主控芯片、cp2102 烧录芯片、lp2992 稳压器、2108A 芯片和 40MHz 晶振、4MB SPI flash，以及一个 mpu6050 芯片和一个 oled 屏幕，并采用板载天线，可以连接蓝牙、WiFi。我们在五根手指上装配了五个弯曲度传感器，其中一个为自制（如图 6），弯曲度传感器实际上是一个高精度的薄片电阻，电阻弯曲后阻值发生变化。根据这个原理，我们使用塑料片、白纸、铅笔芯、电线自制了一个简易的弯曲度传感器，基本可以满足手势识别的需求。我们的手套还配有 400mAh 可充电锂电池，附带一个充电 usb 接口，充电完成后可以大约续航三小时。此外手套上还装配了一个拨动开关和 3D 打印的外壳来放置板子。

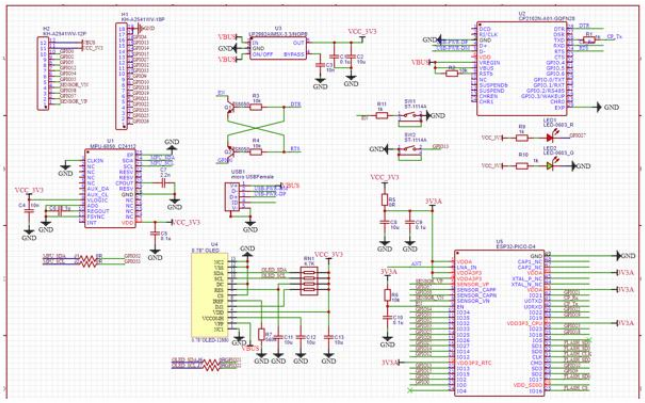

图 3

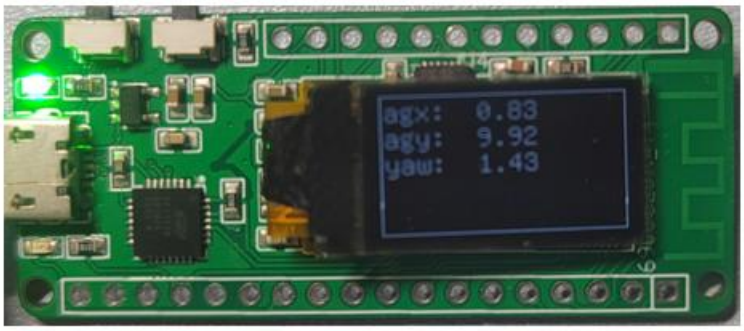

图 4

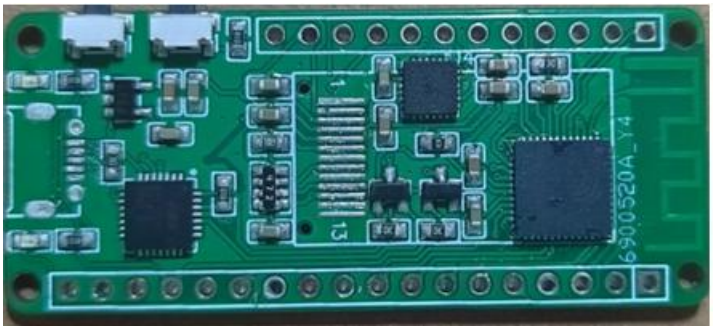

图 5

图 6

4. 核心算法

（1）主程序框架

使用 3 层状态机

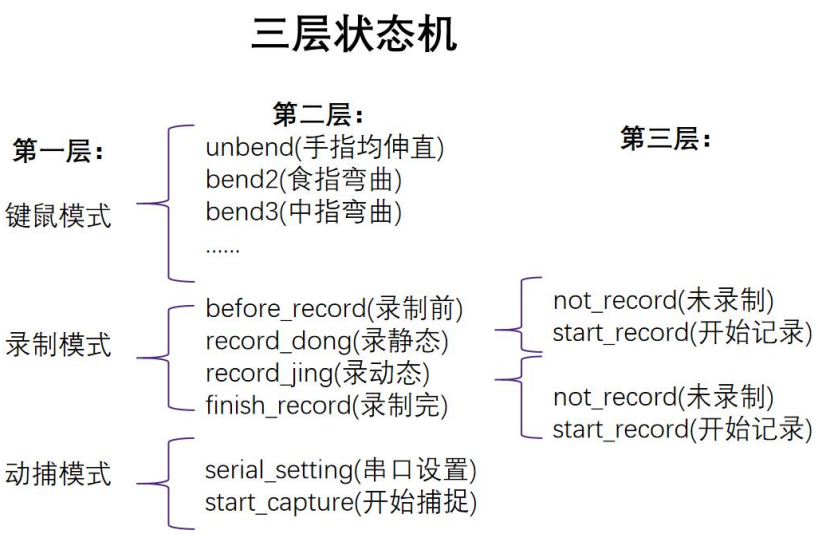

图 7

（2）核心算法

1. 使用滑动窗口滤波和卡尔曼滤波

1.滤波算法：

滑动窗口滤波器

卡尔曼滤波器

```
//计算卡尔曼增益，该增益用于将测量值与当前估计值进行融合，
_kalman_gain = _err_estimate/(_err_estimate + _err_measure);
//计算新的估计值，融合测量值和上一次估计值，以便得到更准确的估计。
_current_estimate = _last_estimate + _kalman_gain * (mea - _last_estimate);
//根据估计值的改进和估计误差，计算新的估计误差，以考虑估计值的不确定性。
_err_estimate =  (1.0 - _kalman_gain)*_err_estimate + fabs(_last_estimate-_current_estimate)*_q;
//将当前估计值保存为上一次的估计值，以便在下一次更新时使用。
_last_estimate=_current_estimate;

return _current_estimate;
```

滤波效果：

图 8

2. 使用朴素贝叶斯分类器和互相关算法

2.手势识别算法：

最终方案：**朴素贝叶斯分类器+(弹性)互相关算法**

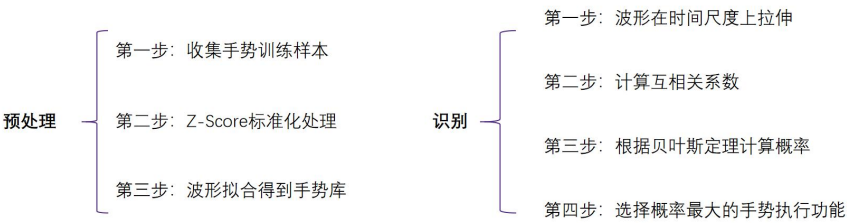

图 9

预处理：第一步：收集手势训练样本；第二步：Z-score 标准化处理；第三步：波形拟合

识别：第一步：波形在时间尺度上拉伸；第二步：计算互相关系数；第三步：根据贝叶斯定

理计算概率；第四步：选择概率最大的手势执行功能

朴素贝叶斯分类器

对于每个手势都有 8 个特征，为 3 轴角度和 5 个手指弯曲度，g_i为某个手势，对于给定的一组特征，其对应g_i手势的概率为：

$$P(g_i \mid pitch, roll, yaw, hand1, \ldots, hand5) = \frac{P(pitch, roll, yaw, \ldots, hand5 \mid g_i) \times P(g_i)}{P(pitch, roll, yaw, \ldots, hand5)}$$

由于 8 个特征之间相互独立，令上式概率为P_i，可写为：

$$P_i = \frac{P(pitch|g_i) \times \ldots \times P(hand5|g_i) \times P(g_i)}{P(pitch, roll, \ldots, hand5)}$$

程序运行时只需要计算P_1、P_2.....P_n中的最大值（且要超过阈值）即认为识别到手势

互相关算法

```
//时间拉伸算法(实现在时间尺度上波形的拉伸或压缩)
void elasticTimeStretch(const float* waveform, int originalLength,
float* stretchedWaveform, int stretchedLength) {
  for (int i = 0; i < stretchedLength; i++) {
    float position = static_cast<float>(i) * (originalLength - 1) / (stretchedLength - 1
    int prevPosition = static_cast<int>(position);
    int nextPosition = prevPosition + 1;
    if (nextPosition >= originalLength) {
      stretchedWaveform[i] = waveform[prevPosition];
    }
    else {
      double fraction = position - prevPosition;
      stretchedWaveform[i] = waveform[prevPosition] + fraction *
      (waveform[nextPosition] - waveform[prevPosition]);
    }
  }
}
```

图 10

```
//计算相似度(相关系数)
float waveSimilarity(const float* wave1, const float* wave2, int size)
{  // 计算均值
    float mean_wave1 = 0.0;
    float mean_wave2 = 0.0;
    for (size_t i = 0; i < size; ++i) {
      mean_wave1 += wave1[i];
      mean_wave2 += wave2[i]; }
    mean_wave1 /=size;
    mean_wave2 /=size;
    float numerator = 0.0;// 计算相关系数的分子和分母
    float denominator_wave1 = 0.0;
    float denominator_wave2 = 0.0;
    for (size_t i = 0; i < size; ++i) {
      float deviation_wave1 = wave1[i] - mean_wave1;
      float deviation_wave2 = wave2[i] - mean_wave2;
      numerator += deviation_wave1 * deviation_wave2;
      denominator_wave1 += deviation_wave1 * deviation_wave1;
      denominator_wave2 += deviation_wave2 * deviation_wave2;
    }  // 计算相关系数
    float correlation_coefficient = numerator / (sqrt(denominator_wave1) * sqrt(denomi
    return correlation_coefficient;
}
```

图 11

3. 其他算法&功能

使用多线程执行程序，将蓝牙串口通信、oled 绘制等函数在新线程中执行，大大缩短主程序执行时间，700 行主程序循环一次仅需 4 毫秒左右；

使用 SPIFFS 文件系统，实现上位机修改后的数据永久储存在开发板上，避免了反复烧录的麻烦。

5. 开发过程

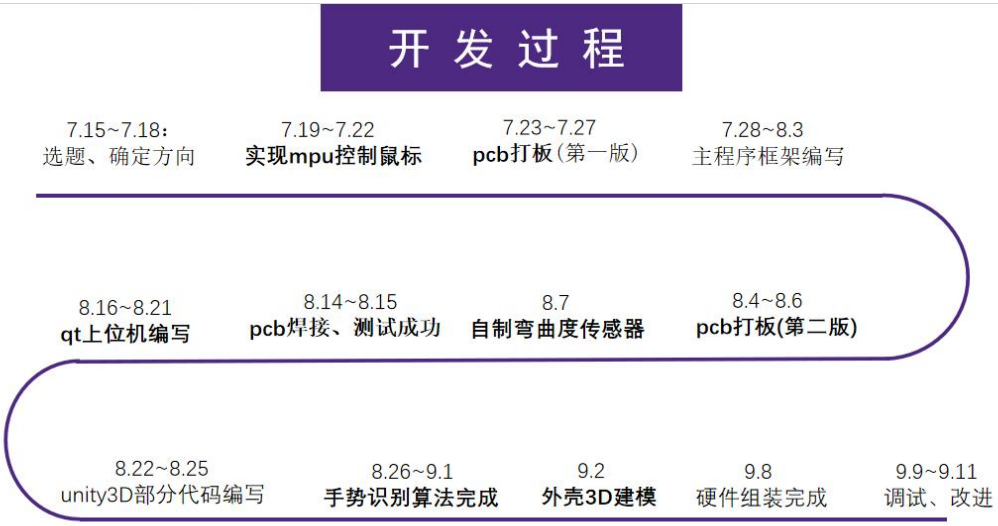

图 12